**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 1 з дисципліни

«Проектування алгоритмів»

„ **Проектування і аналіз алгоритмів зовнішнього сортування**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-12 Боборікин Артем*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2022

# Мета лабораторної роботи

Мета роботи – вивчити основні алгоритми зовнішнього сортування та способи їх модифікації, оцінити поріг їх ефективності.

Варіант 5

1. Псевдокод алгоритму

Random\_gig(); //функція створення файлу

fastSort(); //функція швидкого сортування частин файлу за допомогою ОП

for(n=1 and n<{

write\_files\_b\_c()//розклад файлу А у файли B i C

write\_file\_a() //Запіс файлив B i C у файл А

}

## Програмна реалізація алгоритму

### Вихідний код

1. const fs = ***require***('fs')  
     
   fileSize = 1024\*1024\*100  
   let buffer = ***Buffer***.alloc(1);  
     
   //Запись в файл рандомных чисел  
   function random\_write\_file(num){  
    let rounds = num/1048576  
    let ostatok = num%1048576  
    const fd = fs.openSync("a\_file.bin", 'w');  
    for(let i=0; i<***Math***.floor(rounds); i++){  
    let arr = []  
    for(let j=0; j<1048576; j++){  
    arr.push(***Math***.floor(***Math***.random() \* 256))  
    }  
    let buffer = new ***Buffer***.from( new ***Uint8Array***(arr) );  
    fs.writeSync(fd, buffer, 0);  
    }  
    let arr = []  
    for(let i=0; i<ostatok; i++){  
    arr.push(***Math***.floor(***Math***.random() \* 256))  
    }  
    let buffer = new ***Buffer***.from( new ***Uint8Array***(arr) );  
    fs.writeSync(fd, buffer, 0);  
    fs.closeSync(fd)  
   }  
   //random\_write\_file(fileSize)  
   function random\_gig(){  
    const fd = fs.openSync("a\_file.bin", 'w');  
    let arr = []  
    for(let i=0; i<fileSize; i++){  
    arr.push(***Math***.floor(***Math***.random() \* 256))  
    }  
    let buffer = new ***Buffer***.from( new ***Uint8Array***(arr) );  
    fs.writeSync(fd, buffer, 0);  
    fs.closeSync(fd)  
   }  
   random\_gig();  
     
   //Подготовка файлов  
   fs.writeFileSync('b\_file.bin', '')  
   fs.writeFileSync('c\_file.bin', '')  
   const fd\_a = fs.openSync("a\_file.bin", 'r+');  
   const fd\_b = fs.openSync("b\_file.bin", 'r+');  
   const fd\_c = fs.openSync("c\_file.bin", 'r+');  
     
   //Показать файл  
   function showFile(fd){  
    let arr =[]  
    for(let i=0; i<fileSize; i++){  
    fs.readSync(fd, buffer, 0, 1, i);  
    //console.log(new Uint8Array(buffer)[0])  
    arr.push(new ***Uint8Array***(buffer)[0])  
    }  
    ***console***.log(arr)  
   }  
   showFile(fd\_a)  
     
   ***console***.log('Начало сортировки')  
   let time1 = new ***Date***()  
     
   //Предварительная быстрая сортировка  
   let fastSortNum = ***Math***.pow(2, ***Math***.floor(***Math***.log(fileSize/2)/***Math***.log(2)))  
   let fastArrayLength = fastSortNum<=***Math***.pow(1024, 2)?fastSortNum:***Math***.pow(1024, 2)  
   //let fastArrayLength = 1024  
   //console.log(fastArrayLength)  
   //console.log(Math.log(fastArrayLength)/Math.log(2))  
   for(let i=0; i<=fileSize; i+=fastArrayLength){  
    let fastBuffer = ***Buffer***.alloc(fastArrayLength)  
    fs.readSync(fd\_a, fastBuffer, 0, fastArrayLength, i)  
    if(i+fastArrayLength>fileSize){  
    fastBuffer = ***Buffer***.alloc(fileSize%fastArrayLength)  
    fs.readSync(fd\_a, fastBuffer, 0, fileSize%fastArrayLength, i)  
    let arr = new ***Uint8Array***(fastBuffer)  
    //console.log(arr)  
    arr.sort(compareNumeric)  
    fs.writeSync(fd\_a, arr, 0, fileSize%fastArrayLength, i)  
    break  
    }  
    let arr = new ***Uint8Array***(fastBuffer)  
    //console.log(arr)  
    arr.sort(compareNumeric)  
    fs.writeSync(fd\_a, arr, 0, fastArrayLength, i)  
   }  
   //showFile(fd\_a)  
     
   //Сортировка  
   for(let n=***Math***.log(fastArrayLength)/***Math***.log(2)+1; n<***Math***.ceil(***Math***.log(fileSize)/***Math***.log(2))+1; n++){  
    let finish = false  
    let n\_check\_b=0, n\_check\_c=0  
    let switcher=0, countSwitcher=0  
    for(let i=0; i<fileSize; i+=***Math***.pow(2,n-1)){  
    let read\_buffer  
    if(i+***Math***.pow(2,n-1)>fileSize){  
    read\_buffer = ***Buffer***.alloc(fileSize-i)  
    fs.readSync(fd\_a, read\_buffer, 0, fileSize-i, i);  
    finish = true  
    }else{  
    read\_buffer = ***Buffer***.alloc(***Math***.pow(2,n-1))  
    fs.readSync(fd\_a, read\_buffer, 0, ***Math***.pow(2,n-1), i);  
    }  
     
    let arr\_a = new ***Uint8Array***(read\_buffer)  
    let arr\_b=[], arr\_c=[]  
    for(let one of arr\_a){  
    if(countSwitcher===***Math***.pow(2, n)/2){  
    countSwitcher=0  
    switcher=!switcher  
    }  
    if(switcher){  
    arr\_b.push(one)  
    }else{  
    arr\_c.push(one)  
    }  
    countSwitcher++  
    }  
    // console.log('ARR\_B: ', arr\_b)  
    // console.log('ARR\_C: ', arr\_c)  
    // console.log('-------')  
    if(arr\_b.length>0){  
    fs.writeSync(fd\_b, new ***Buffer***.from( new ***Uint8Array***(arr\_b) ), 0, arr\_b.length, n\_check\_b);  
    n\_check\_b+=arr\_b.length  
    }  
    if(arr\_c.length>0){  
    fs.writeSync(fd\_c, new ***Buffer***.from( new ***Uint8Array***(arr\_c) ), 0, arr\_c.length, n\_check\_c);  
    n\_check\_c+=arr\_c.length  
    }  
    if(finish)break  
    }  
     
    let b\_length = n\_check\_b, c\_length = n\_check\_c  
    n\_check\_b=0; n\_check\_c=0  
    for(let i=0, b\_i=0, c\_i=0; i<fileSize; ){  
    let write\_buffer\_b = ***Buffer***.alloc(***Math***.pow(2, n-1))  
    let write\_buffer\_c = ***Buffer***.alloc(***Math***.pow(2, n-1))  
     
    let array\_b = [], array\_c = []  
     
    if(b\_i+***Math***.pow(2, n-1)<=b\_length){  
    fs.readSync(fd\_b, write\_buffer\_b, 0, ***Math***.pow(2, n-1), b\_i);  
    }else{  
    if(b\_length-b\_i!==0){  
    write\_buffer\_b = ***Buffer***.alloc(b\_length-b\_i)  
    fs.readSync(fd\_b, write\_buffer\_b, 0, b\_length-b\_i, b\_i)  
    }  
    }  
    if(c\_i+***Math***.pow(2, n-1)<=c\_length){  
    fs.readSync(fd\_c, write\_buffer\_c, 0, ***Math***.pow(2, n-1), c\_i);  
    }else{  
    if(c\_length-c\_i!==0){  
    write\_buffer\_c = ***Buffer***.alloc(c\_length-c\_i)  
    fs.readSync(fd\_c, write\_buffer\_c, 0, c\_length-c\_i, c\_i)  
    }  
    }  
    if(b\_length-b\_i!==0){  
    array\_b = new ***Uint8Array***(write\_buffer\_b)  
    b\_i+=array\_b.length  
    }  
    if(c\_length-c\_i!==0){  
    array\_c = new ***Uint8Array***(write\_buffer\_c)  
    c\_i+=array\_c.length  
    }  
     
    let arr = [...array\_b, ...array\_c]  
    arr.sort(compareNumeric)  
     
    //console.log('Sorted Array: ', arr)  
     
    fs.writeSync(fd\_a, new ***Buffer***.from(new ***Uint8Array***(arr)), 0, arr.length, i);  
    i+=arr.length  
    }  
     
    //console.log('AAA')  
    //showFile(fd\_a)  
    fs.writeFileSync('b\_file.bin', '')  
    fs.writeFileSync('c\_file.bin', '')  
   }  
     
     
   let time2 = new ***Date***()  
   ***console***.log('Сортировка пройдена')  
   showFile(fd\_a)  
   ***console***.log('Время: ', time2-time1)  
     
   //Закрытие потоков  
   fs.close(fd\_a)  
   fs.close(fd\_b)  
   fs.close(fd\_c)  
   fs.unlinkSync('b\_file.bin')  
   fs.unlinkSync('c\_file.bin')  
     
     
   function compareNumeric(a, b) {  
    if (a > b) return 1;  
    if (a === b) return 0;  
    if (a < b) return -1;  
   }

Висновок

При виконанні роботи я оволодів знаннями алгоритмів зовнішнього сортування. В моєму варіанті я використовував метод прямого злиття, якій у базовому вигляді займав пріблізно 1.8 хвилини для сортування 1мб. Після цього я провів модифікації алгоритму та отримав результат 300мс. Я провів 2 модифікації:

1 – використання додаткової оперативної пам’яті під час зчитування та запису даних у файл (у місці об’єднаних комірок) Максимальна кількість додаткової оп – 60 байтів під час кожного використання (залежить від розміру масиву – степінь двійкі інколи помножена на 2).

2 - використання додаткової оперативної пам’яті під час попереднього сортування масиву. Алгоритм частинами сортує файл у ОП.